Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Žiga Strgar

Interno orodje za nadzor poslovnih procesov projekta za souporabo vozil

diplomsko delo

visokošolski strokovni študijski program
prve stopnje
računalništvo in informatika

Mentor: prof. dr. Marko Bajec
Somentor: dr. Matej Grošelj

Ljubljana, 2019

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo: Interno orodje za nadzor poslovnih procesov projekta za souporabo vozil

Tematika naloge:

V okviru diplomskega dela opišite področje souporabe vozil v Sloveniji. Na tej osnovi izdelajte zasnovo in računalniško orodje, ki bo v pomoč pri nadzoru vseh procesov, ki se v okviru souporabe vozil izvajajo. Opišite načrt rešitve ter rešitev predstavite.

Iskreno zahvalo izrekam mentorju diplomske naloge prof. dr. Marku Bajcu in somentorju dr. Mateju Grošlju za ves vloženi čas, strokovno vodenje in dane nasvete, ki so prispevali k dokončanju te diplomske naloge.

Zahvalil bi se tudi svoji družini, ki mi je omogočila pridobitev željene izobrazbe, me spodbujala, usmerjala in učila na vseh korakih življenja.

Podjetju Avant car d.o.o. za omogočen vpogled v delovanje projekta souporabe vozil Avant2Go.

Vsem, ki me motivirate, podpirate, svetujete in pomagate.

Mojemu očetu.

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
API	Application Programable Interface	aplikacijski programski vmesnik
ARSO	Slovenian Environment Agency	Agencija Republike Slovenije za okolje
CO_2	Carbon dioxide	ogljikov dioksid
CSS	Cascading Style Sheets	prekrivni slog
DOM	Document Object Model	objektni model dokumenta
EAP	Early Access Program	program zgodnjega dostopa
HTML	HyperText Markup Language	jezik za označevanje nadbesedila
JSON	JavaScript Object Notation	notacija objektov JavaScript
KPI	Key Performance Indicator	ključni kazalnik uspešnosti
MVC	Model-View-Controller	model-pogled-logični krmilnik
MQ	Message Queue	sporočilna vrsta
ORM	Object-Relational Mapping	objektno-relacijska preslikava
PHP	PHP: Hypertext Preprocessor	PHP: predprocesor nadbesedila
PSR	PHP Standards Recommendations	priporočeni standardi PHP
SDK	Software Development Kit	paket za razvoj programske opreme
SQL	Structured Query Language	strukturiran povpraševalni jezik

Povzetek

Naslov: Interno orodje za nadzor poslovnih procesov projekta za souporabo vozil

Avtor: Žiga Strgar

Poslovno domeno pričujočega diplomskega dela predstavlja sistem souporabe 100 % električnih vozil, ki predstavlja 24/7 samopostrežno možnost najema vozil preko aplikacije na pametnem telefonu. Pri spremljanju razvoja sistema sem identificiral naslednji problem: z večanjem števila uporabnikov, vključenih vozil in prevzemno-vračilnih lokacij ter funkcionalnosti se veča tudi njegova kompleksnost, pri čemer morajo upravljalci storitve na različnih ravneh z namenom učinkovitega upravljanja ohraniti nadzor in vpogled v ključne parametre delovanja sistema. Cilj diplomskega dela je predstavitev rešitve, nadzorne plošče, ki podjetju in upravljalcem omogoča natančen vpogled v delovanje sistema ter na pregleden in enostaven način prikazuje ključne parametre sistema glede na raven in potrebe upravljalca, s čimer je omogočena tudi hitrejša odzivnost in zaznavanje morebitnih odstopanj. Pri doseganju cilja sem uporabil sodobne tehnologije za razvoj spletnih aplikacij ter uveljavljene komunikacijske protokole. Rešitev sem tudi analiziral in predstavil možnosti za prihodnje nadgradnje na tehničnem nivoju in na nivoju poslovne domene.

Ključne besede: nadzorna plošča, poslovni procesi, pregled, aplikacija v realnem času, suporaba vozil.

Abstract

Title: A tool for controlling business processes within the car sharing project

Author: Žiga Strgar

The business domain of this final paper presents a sharing system of 100% electric vehicles, which offers a 24/7 self-service vehicle rental through a smart phone application. In monitoring the system development, I identified the following problems: by increasing the number of users, included vehicles, pick-up and drop-off locations, and its functionality, the system's complexity also increases, which means the service managers have to maintain the control and insight into key parameters of the system operation in order to manage it effectively. The goal of this final paper is to present a solution—a control panel that provides the company and system managers with an accurate insight into the system operation, and a transparent and simple display of the system's key parameters according to the user's level and requirements, which also enables a faster responsiveness and detects potential deviations. In achieving the goal, I used modern web application development technologies and established communication protocols. I analysed the solution and presented options for future upgrades on the technical and business domain level.

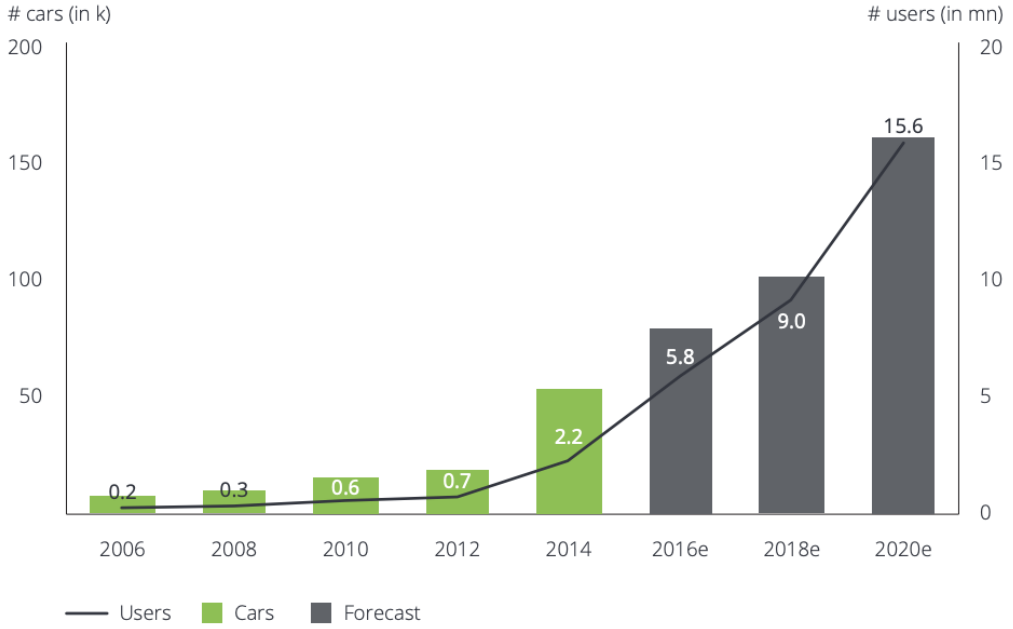
Keywords: control panel, business processes, insight, real-time application, vehicle sharing.

Poglavje 1

Uvod

1.1 Poslovna domena

Souporaba vozil se v svetu pojavlja kot nov inovativen način, kako potrošniki uporabljamo avtomobile. S tem souporaba vozil širi svoje prednosti mobilnosti na posameznika, ne da bi ta nosil ceno lastništva in oskrbe vozila [3]. Za lastniški avtomobil lastnik namreč plača celoten strošek, čeprav je le-ta v uporabi zgolj 10 % časa. Gledano dolgoročno nam vsako vozilo v okviru delitvene ekonomije souporabe vozil znižuje število lastniških avtomobilov, posledično pa to prinese manj vozil na cesti, manj hrupa in do 100 gramov manj izpustov CO_2 na prevoženi kilometer na vozilo. V Evropi je takšen način mobilnosti priljubljen predvsem v Nemčiji, Franciji, Italiji, Švici in skandinavskih državah. O tem, kako hitro raste sama storitev souporabe vozil, priča tudi slika 1.1 [3].



Slika 1.1: Rast souporabe vozil v Evropi in napoved do leta 2020 [3]

Souporaba vozila predstavlja alternativo klasičnemu lastništvu vozila, pri čemer uporabniki uporabljajo vozila le takrat, ko jih potrebujejo in za uporabo vozila samo takrat tudi plačajo [4]. Hitrejšo rast beleži v urbanih okoljih [4]. Višja uveljavljenost in povpraševanje prinašata čedalje večjo kompleksnost povezav med sistemi ob zasledovanju čim boljše uporabniške izkušnje.

1.2 Struktura diplomskega dela

Diplomsko delo začnemo z opisom problema in rešitve, sledi načrtovanje vidikov rešitve, kot so moduli (opozorila o težavnih vozilih in lokacijah, statusi sistemov, neaktivna vozila, grafi, poročila o uporabi sistema idr.) in upravljalci, komunikacija aplikacije z drugimi storitvami in ključni primeri uporabe. Nadaljujemo z opisom uporabljenih orodij in tehnologij ter prikažemo rešitev. Za konec predlagamo še možnosti nadaljnjega razvoja s stališča tehnologije in poslovne domene projekta.

Poglavje 2

Problem in rešitev

2.1 Problem

V Sloveniji se je leta 2016 začel projekt Avant2Go, ki ponuja točno takšno storitev v trenutno štirih mestih po Sloveniji, dosegljiva pa je vsak dan in to cel dan [1]. Na začetku je bilo na voljo 12 odprtih lokacij s 30 vozili. Danes, v času pisanja tega diplomskega dela, pa je v sistemu vključenih že več kot 150 vozil in 62 različnih prevzemno-vračilnih lokacij v Ljubljani, Mariboru, Kranju in Murski Soboti [2]. To v primerjavi z začetkom pomeni kar 500-odstotno povečanje razpoložljive flote vozil in 517-odstotno povečanje števila lokacij. Z večanjem števila uporabnikov, vključenih vozil in prevzemno-vračilnih lokacij ter funkcionalnosti se veča tudi kompleksnost sistema, pri čemer morajo upravljalci storitve na različnih ravneh z namenom učinkovitega upravljanja ohraniti nadzor in vpogled v ključne parametre delovanja sistema.

2.2 Rešitev

Izhajajoč iz opredeljene poslovne domene in predstavljenega problema diplomskega dela v nadaljevanju predstavljamo rešitev v obliki nadzorne plošče (spletne aplikacije), ki podjetju in upravljalcem omogoča natančen vpogled v delovanje sistema ter na pregleden in enostaven način prikazuje ključne pa-

rametre sistema glede na raven in potrebe upravljalca, s čimer je omogočena tudi hitrejša odzivnost na morebitna odstopanja. Rešitev je namenjena vsem ravnem, od operativnega dela logistike do vodje logistike, računovodstva, centra za podporo uporabnikom in vodstva podjetja.

Oseba, ki je zaposlena v logistiki, lahko preko aplikacije spremlja spremembe lokacij in avtomobilov, ki predstavljajo težavo. Enake zadeve lahko vidi oseba, zaposlena kot pomoč uporabniku, da lahko v primeru klica stranke asistira s podanim znanjem o lokacijah in avtomobilih. Računovodstvo ima učinkovit pregled nad vsemi transakcijami, ki jim lahko sledi in posreduje v primeru težav. Vodstvo ima pregled nad celotnim sistemom, vključno z logistiko in računovodstvom, z dodatkom pregleda konkretnih številčnih kazalnikov, ki nakazujejo na finančno uspešnost projekta. Ob tem ima možnost vpogleda v trende rasti, izračun utilizacije lokacij in sistema ter generiranja poročil uporabe.

Poglavje 3

Načrtovanje razvoja rešitve

3.1 Funkcionalne zahteve

Rešitev mora ustrezati različnim funkcionalnim zahtevam, razdeljenim na funkcionalne module, ki so na voljo v zgoščeni obliki kot kazalniki uspešnosti za upravljalca, in ravni upravljalcev, ki dostopajo do modulov.

Aplikacija mora podpirati:

- **Prikaz različnih kazalnikov delovanja storitve** – Slednji so predstavljeni s številčno vrednostjo.
- **Seznami različnih dolžin** – Predstavljajo potrebne podatke.
- **Grafi** – Vpogled v graf preko različnih časovnih intervalov vzorčenja.
- **Živi tok podatkov** – Prikazujejo opis dogodka, skupaj s časovno oznako.

Naštete podprte funkcionalne zahteve potrebujejo osveževanje v realnem času, pri čemer želimo tudi natančen vpogled v različne trende in zgodovino poslovanja. Slednja dva načina se izvajata na zahtevo, torej ne potrebujeta konstantnega osveževanja v realnem času.

V nadaljevanju opredeljujemo posamezne module:

- **Prazne in polne prevzemno-vračilne lokacije** – Seznam praznih in polnih prevzemno-vračilnih lokacij, skupaj s številom razpoložljivih vozil in številom vseh parkirnih mest na posamezni lokaciji.

- **Prazna in umazana vozila** – Seznam vozil s prenizko ravnijo električne energije in seznam vozil, ki imajo ocenjeno čistočo (ob končanju vsakega najema oseba subjektivno oceni čistočo vozila) pod sprejemljivo vrednostjo. Izpis, iz katerega upravljalec vidi registrsko številko vozila, lokacijo vozila in pripadajočo težavno številko (raven energije, ocena čistoče).

- **Trenutno aktivni zaposleni v klicnem centru in logistiki** – Seznam ljudi, ki so trenutno na voljo in opravljajo svojo obveznost bodisi v logistiki ali v klicnem centru za pomoč strankam.

- **Živi tok podatkov različnih možnosti kontakta pomoči strankam** – Živi tok podatkov, predstavljen kot seznam dogodkov s časovno oznako in opisanim dogodkom. Viri obsegajo klicni center, pomoč preko e-pošte, tekstovni pogovor v živo.

- **Status sistema** – Spremljanje statusov različnih sistemov, od katerih je odvisen projekt. Z vidno vizualno spremembo ob zaznani težavi (primer: nedosegljiva storitev, sporočanje težav idr.).

- **Prevzemno-vračilne lokacije** – Spremljanje statusa vsake lokacije, dosegljive v sistemu in njen prikaz po pripadajočih regijah. Potrebna je tudi vidna vizualna sprememba ob zaznani težavi (prazna ali polna lokacija).

- **Živi tok opravljenih transakcij** – Živi tok podatkov, sestavljen kot seznam dogodkov, označenih s časovno oznako in opisom transakcije. Zraven dodan tudi tip transakcije (kreditna kartica, osebni račun, poslovni račun) in unikatni kazalnik iz plačilnega sistema za lažje sledenje transakciji.

- **Neaktivna vozila** – Seznam vozil, ki so že dlje časa neaktivna. Tu gre za odsotnost osvežitve geo-lokacije vozila ali odsotnost aktivne rezervacije.

- **Pregled po regijah delovanja sistema** – Za vsako regijo, kjer sistem deluje, so prikazane pripadajoče lokacije z dodatnimi kazalniki, kot so seznam umazanih in praznih avtomobilov, število najemov (logistike in strank), število rezerviranih vozil, število aktivnih najemov in število prostih vozil.

- **Grafi** – Predstavitev določenih kazalnikov z grafom. Utilizacija vozil (vzorčenje na 15 minut), število novih uporabnikov, prihodek na vozilo, povprečna dnevna utilizacija. Zadnji trije imajo vsi predstavljene dnevne vrednosti.

- **Različni kazalniki predstavljeni s številko** – Številčni kazalniki, ki prikazujejo bodisi trenutno stanje ali dnevno stanje. Slednji zajemajo število trenutno aktivnih najemov, število trenutno rezerviranih vozil (vozilo ni vidno v sistemu in čaka na prevzem stranke), število prostih vozil, rezervacije strank in logistike, število novo registriranih uporabnikov, število prevzemno-vračilnih lokacij, opravljenih transakcij glede na tip plačilne metode (kreditna kartica, osebni račun, poslovni račun), povprečno vrednost transakcije, število vseh transakcij, aktivne uporabnike, število potrjenih uporabnikov, število uporabnikov, ki jim je onemogočen dostop do uporabe storitve, število podjetij, ki uporabljajo storitev, število parkirnih mest (javnih in poslovnih), število upokojenih vozil, število vozil glede na model vozila in skupni znesek transakcij. Nekateri izmed kazalnikov obsegajo tudi različna časovna obdobja, medtem ko nekateri prikazujejo realno stanje ali celo stanje trenutnega dneva (ponastavitev ob polnoči).

- **Spremljanje trendov** – Vpogled v sistem za poljubno izbrano časovno obdobje parametrov, kot so graf prihodka in števila najemov, graf pri-

hodkov glede na lokacijo/model vozila/regijo, število registracij novih uporabnikov in graf utilizacije prihodka glede na model.

- **Vpogled v registrirane uporabnike in aktivna vozila** – Prikaz in filtriranje po uporabnikih in vozilih z omejenimi prikazanimi atributi.
- **Utilizacija lokacij** – Prikaz utilizacije prevzemno-vračilnih lokacij sistema. Merjenje dolžine, ko je bila lokacija bodisi polna ali prazna in nato čas, izražen z odstotkom.
- **Poročila o uporabi** – Generiranje poročil o uporabi sistema na dnevni, mesečni in letni osnovi. Medtem ko se vse našteto pošilja samodejno preko e-pošte, mora biti enak vir podatkov dosegljiv na zahtevo tudi preko uporabniškega vmesnika.

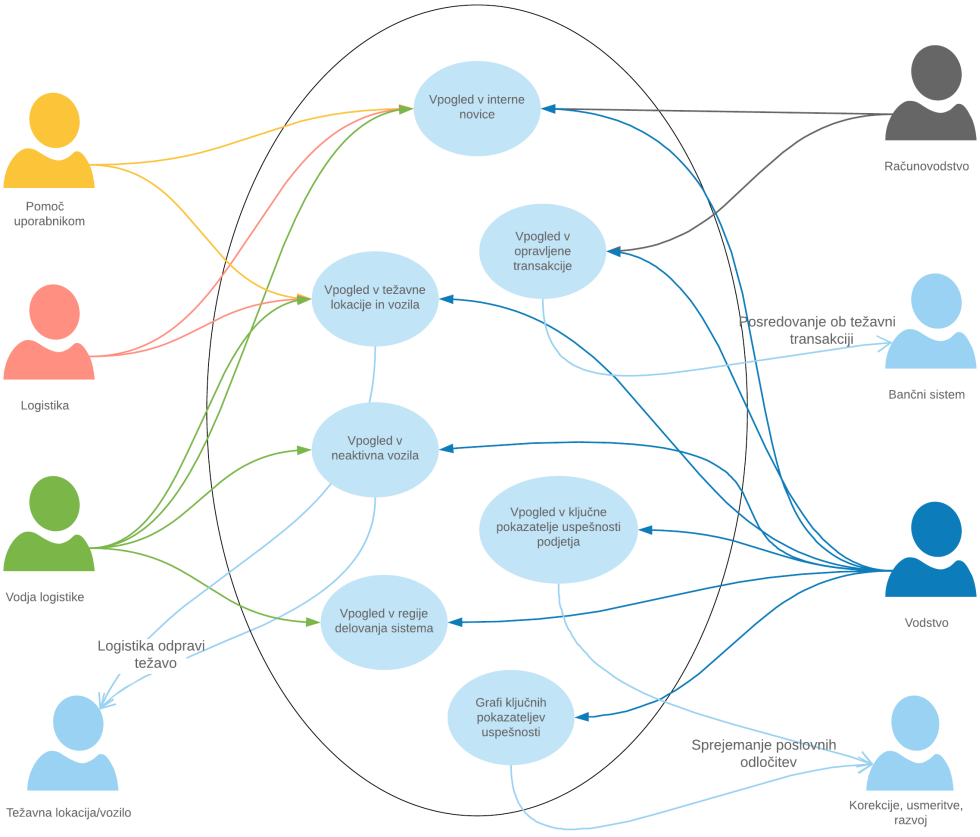
3.2 Ravni upravljalcev

Upravljalec dobi glede na svoje zadolžitve tudi dostop do modulov (poglavje 3.1), ki mu pomagajo pri učinkovitejšem izvajanju delovnih zadolžitev. Z avtorizacijo se vpogled omeji na podatke, ki jih potrebuje posamezna raven upravljalcev. Natančen pregled, kako so razdeljene pravice upravljalcem, je prikazan tudi na sliki 3.1. Sistem vključuje naslednje ravni upravljalcev:

- **Pomoč uporabnikom** – Upravljalec oziroma zaposleni v oddelku pomoči uporabnikom in ima dodeljen dostop do vpogleda v težavna vozila in lokacije.
- **Logistika** – Upravljalec v logistiki ima pravico vpogleda v težavne lokacije in vozila.
- **Vodja logistike** – Medtem ko ima vodja logistike enake pravice kot upravljalec *Logistika*, ima poleg tega še dodatno možnost pregleda neaktivnih vozil in posameznih regij delovanja sistema. Ob tem ima vodja logistike pravico spremljanja trendov, generiranja poročila o uporabi in nadzora nad utilizacijo lokacij.

- **Računovodstvo** – Računovodski oddelek podjetja ima vpogled v seznam vseh opravljenih transakcij (uspešnih in neuspešnih), ob tem pa še dodanih nekaj drugih kazalnikov, kot so število opravljenih transakcij glede na izbrani način plačila (kreditna kartica, osebni račun, poslovni račun), vrednost povprečnega najema in število vseh opravljenih transakcij.

- **Vodstvo** – Vodstvo podjetja ima vpogled v vse veje sistema souporabe vozil in tako lahko dostopa do popolnoma vseh strani. Za njegove potrebe je poleg že prej omenjenih modulov omogočen tudi pregled grafov in vseh ključnih kazalnikov uspešnosti.



Slika 3.1: Shema dostopov glede na raven upravljalca in funkcionalne zahteve

Slika 3.1 prikazuje dodeljene pravice posameznih uporabnikom nadzorne plošče (upravljalcev sistema souporabe vozil) glede na njihove poslovne potrebe in delovne dejavnosti (različne barve predstavljajo različne ravni upravljalcev). Omogoča dejavno spremljanje posameznih deležnikov na različnih ravneh. Na primer, vodja logistike ima možnost v realnem času spremljati težavne prevzemno-vračilne lokacije (lokacije, ki so bodisi povsem polne bodisi povsem prazne), na osnovi česar logistična ekipa poskrbi za vnovično vzpostavitev želenega stanja na področju prevzemno-vračilnih lokacij (vsaka lokacija ima vsaj eno razpoložljivo vozilo za najem in vsaj en prostor za vrnitev vozila). Vodstvo ima na osnovi spremljanja KPI-jev možnost nadzora nad širšo sliko delovanja projekta, s čimer lahko po potrebi sprejema korektivne ukrepe in/ali podaja usmeritve drugim udeležencem (upravljalcem) ter sprejema razvojne odločitve v okviru projekta souporabe vozil. Glede na njihove poslovne potrebe imajo tudi drugi upravljalci (na primer center za pomoč uporabnikom, računovodstvo idr.) s spremljanjem posameznih dodeljenih modulov osnovo za učinkovitejše odločanje pri dnevnem delovanju. Slika 3.1 zaradi preglednosti ne prikazuje povezav z akterjem (*angl. cron*), ki skrbi da so zgoraj navedene funkcionalnosti z izjemo "Grafi ključnih pokazateljev uspešnosti" vedno oskrbovane z zadnjimi relevantnimi informacijami, ki so nato predstavljene upravljalcem.

3.3 Tehnične zahteve

V tem podpoglavju so opisane tehnične zahteve rešitve z vidika implementacije in same uporabe pri izvajanju službenih dejavnosti. Zahteve so naslednje: prikaz podatkov v realnem času, avtentikacija in avtorizacija, prilagojenost televizijskim zaslonom, odzivni čas, razširljivost, modularnost in uporaba zahtevane predloge.

- **Zahtevana predloga** – Uporaba odprtokodne rešitve belgijskega podjetja Spatie [14], s čimer so se deloma določile tudi uporabljene tehnologije, ki obsegajo Laravel za zaledni del sistema in Vue.js na čelnem delu,

ki s svojim reaktivnim sistemom DOM omogoča prikaz novih podatkov brez osveževanja strani.

- **Prikaz podatkov v realnem času** – Vsi podatki morajo biti upravljalcu dostopni na nadzorni plošči v realnem času, in sicer v prilagojeni obliki kot kazalniki uspešnosti, predstavljeni na pregleden in enostaven način.

- **Avtentikacija** – Rešitev predstavlja interno orodje, ki ni namenjeno širši javnosti. Za dostop do nadzorne plošče potrebujemo veljavno kombinacijo uporabniškega imena in gesla.

- **Avtorizacija** – Glede na občutljivost nekaterih podatkov, predstavljenih na sami nadzorni plošči in glede na sam spekter zadolžitev posameznega upravljalca se mu tako glede na določene službene obveznosti omeji tudi dostop do drugih modulov.

- **Prikaz na televizijskem zaslonu** – Sam način predstavitve podatkov je zamišljen kot na enem velikem zaslonu v pisarni, kjer imajo vsi upravljalci dostop do podatkov, ki so jim namenjeni. S tem morata biti vizualno prilagojeni tudi sama vsebina in oblika prikaza. Vsi elementi nadzorne plošče morajo biti tako 100-odstotno vidni in prisotni v vsakem trenutku.

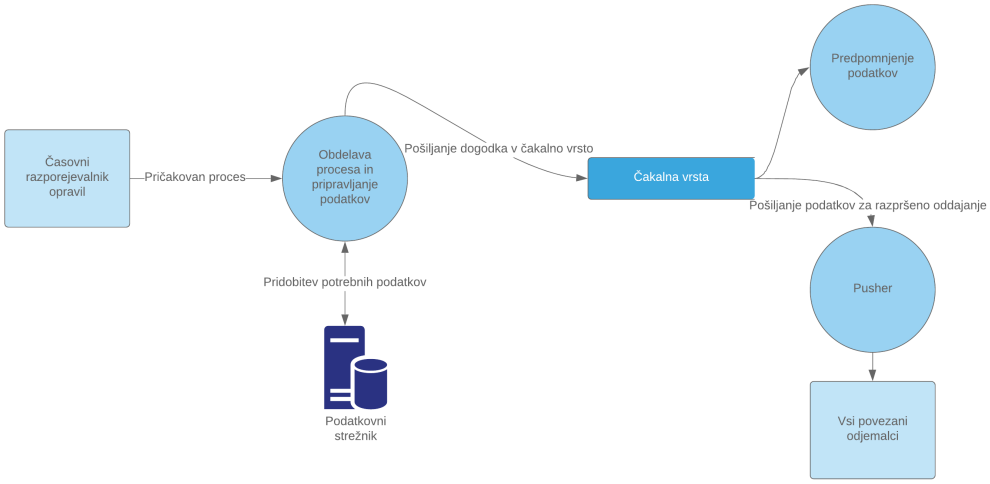
- **Odzivni čas** – Omejitev časovnega razporejevalnika predstavlja dejstvo, da lahko zgolj na vsako minuto izvede določen ukaz, kar je popolnoma sprejemljivo za potrebe aplikacije, vendar mora sama aplikacija te obdelane podatke dostaviti vsem odjemalcem v časovnem razmiku 10 sekund od izvršitve ukaza.

- **Razširljivost** – Novi modul mora biti relativno hitro dostavljen, saj odjemalec na zaledni strani ni tesno povezan s poslovno logiko podjetja (načelo Open/Closed). Slednje omogoča nadzorni plošči pridobivanje informacij iz drugih virov na enostaven in učinkovit način.

- **Modularnost** – Od vsakega modula se zahteva popolna modularizacija funkcionalnosti, kar pomeni, da noben modul ne sme biti v svojem delovanju odvisen od drugega, ne pri pridobivanju, agregiranju ali prikazovanju podatkov (kar vsebuje skupni prikaz primera: seznam prikaza polnih prevzemno-vračilnih lokacij ni odvisen od prikaza praznih prevzemno-vračilnih lokacij, to pomeni, da lahko stoji na strani kot samostojen element).

3.4 Ključni primeri uporabe

Rešitev je sestavljena iz dveh delov. Prvi del je nadzorna plošča, drugi del pa predstavlja sistem za poročanje in preverjanje. To v nadaljevanju prikazujemo s primeroma uporabe.

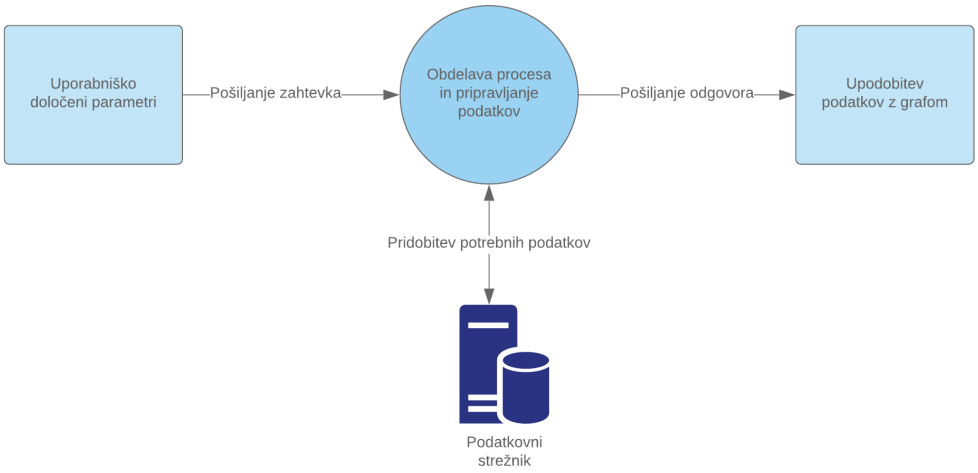


Slika 3.2: Primer uporabe nadzorne plošče

Na sliki 3.2, kjer ključno vlogo igra časovni razporejevalnik opravil, vidimo, da le-ta sproži izvajanje procesa, ki pridobi in obdela podatke v smiselno celoto, ki jo nato posreduje naprej po cevovodu, kjer počakajo na izvajanje in kasnejše predpomnjenje ter razpršeno oddajanje. Slednje omogoča želeno uporabniško izkušnjo, da upravljalec na svojem delovnem mestu ne osveži strani za prikaz novih podatkov, temveč je ta videna slika vedno realno stanje sistema.

V ozadju, ki na sliki 3.2 zaradi večje preglednosti ni predstavljeno, se vsako minuto izvede določen del sistema Laravel (opisan v poglavju 4.1), kjer so zapisana pravila o pogostosti izvajanja določenega opravila (primer: vsako minuto pridobi podatke o stanju lokacij, iz česar vidimo stanje prevzemno-vračilne lokacije). Vsa prihajajoča stanja so že opisana in predstavljena na sliki 3.2. Pri koraku predpomnjenja podatkov, ki se zgodi

po privzeto nastavljenem načinu znotraj samega ogrodja Laravel, sledi še sočasno sporočanje vsem odjemalcem, pri čemer odjemalec prejme agregirane podatke v pričakovani obliki in novo stanje upodobi na zaslonu.

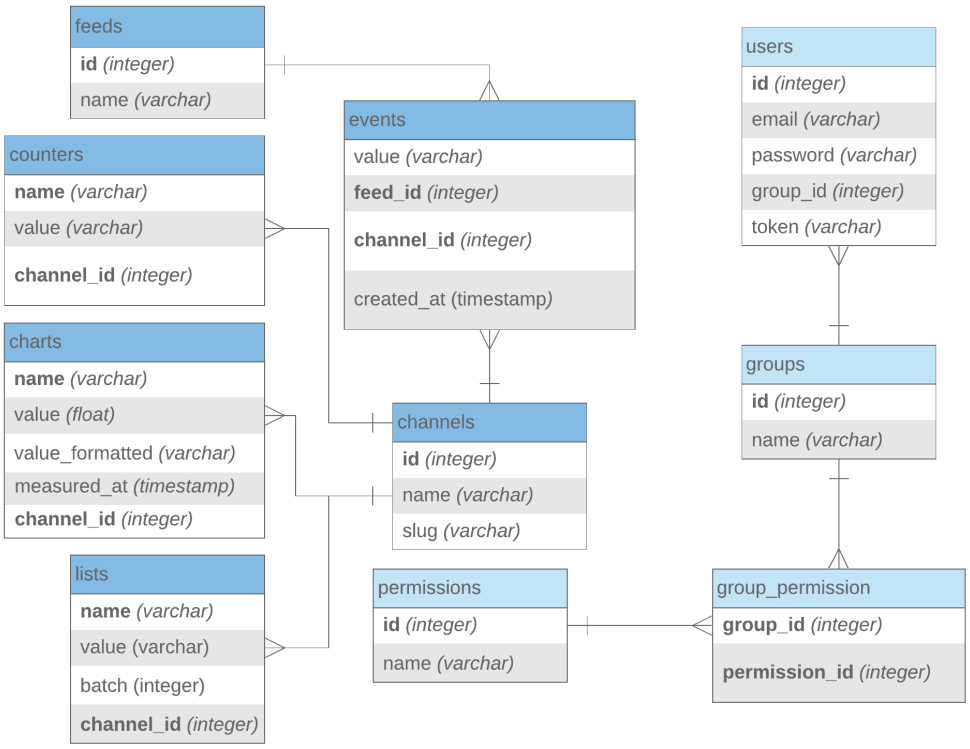


Slika 3.3: Primer uporabe sistema za nadzor in poročanje

Slika 3.3 prikazuje prožen sistem, kjer lahko upravljalec glede na vsakokratne potrebe izbira med različnimi kombinacijami filtrov (na primer prihodek na izbrano lokacijo v izbranem časovnem obdobju), s čimer dobi podrobnejšo sliko o delovanju projekta. Upravljalec tako s kliki na različna stikala in vnosna polja, ki predstavljajo filtre, ustvari opis želenega stanja. Nato s klikom na gumb pošlje asinhroni zahtevek, skupaj s filtri za pridobitev omenjenega stanja. Strežnik ta zahtevek prilagodi smernicam, predstavljenim s strani zalednega sistema in mu pošlje zahtevek za želene podatke. Ob uspešni pridobitvi obdela podatke v obliko, ki jo odjemalec upravljalca pričakuje in mu jih vrne kot odgovor na njegovo poizvedbo. S temi podatki se nato na strani upravljalca v brskalniku izriše želeno stanje v obliki grafa.

3.5 Podatkovni model

Zgornja dva ključna primera uporabe uporabljata enak vir podatkov (zaledni sistem projekta), vendar v primeru, opisanem na sliki 3.3, zadaj ni podatkovnega modela (slika 3.4), temveč aplikacija z vsakim zahtevkom na novo pridobi podatke. Medtem ko iz prvega primera, opisanega na sliki 3.2, aplikacija uporabi podatkovni model za prikaz podatkov ob prihodu na stran, kasneje pa posodobljene podatke dobi preko sistema za sočasno oddajanje.



Slika 3.4: Podatkovni model rešitve

Slika 3.4 prikazuje podatkovni model, kjer je razvidna struktura uporabnikov s pripadajočimi skupinami (ravni upravljalcev), ki jim dodelimo pravice do posameznih modulov. Poleg tega je predstavljena tudi struktura hranjenja podatkov nadzorne plošče, kjer uporabljamo osveževanje v realnem času in predpomnjenje podatkov. Najpomembnejšo točko predstavlja

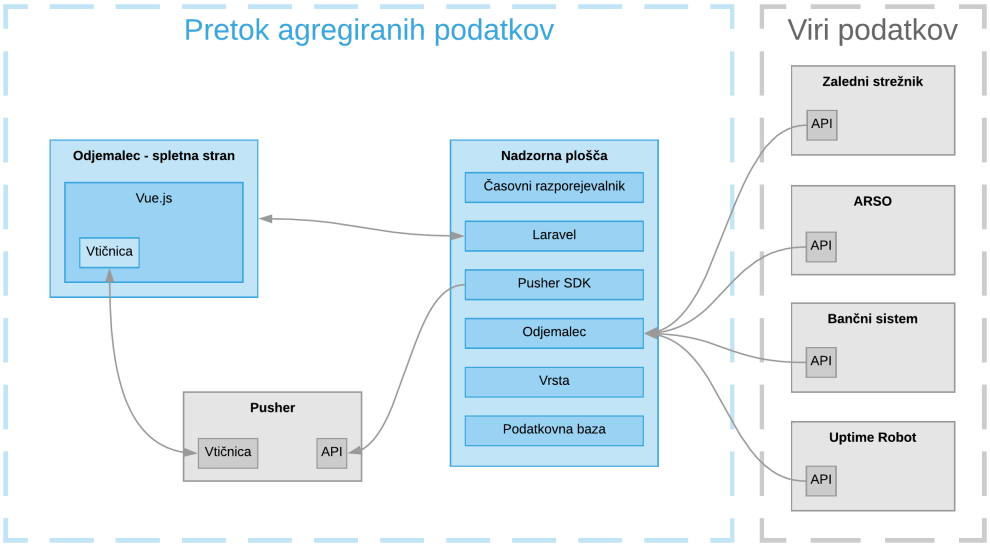
tabela *channels*, preko katere so povezani vsi podatki za prikaz na nadzorni plošči. Vse tabele, ki hranijo različne oblike podatkov, imajo kot primarni ključ določeno ime samega dogodka, preko katerega se sočasno oddajajo poslani podatki. Primarni ključ sestavlja tudi tuji ključ, podedovan iz tabele *channels*, kjer so predstavljeni splošni kanal oddajanja in ločeni kanali po regijah obratovanja sistema.

3.6 Arhitektura sistema

V sklopu diplomskega dela smo razvijali:

- **Strežniška aplikacija** – Vmesnik med odjemalcem in zalednim sistemom neobdelanih podatkov, imenovan nadzorna plošča.
- **Odjemalec - spletna stran** – Sestavljena iz različnih komponent.

V poglavju opisujemo, kako se med sistemi prenašajo podatki in vse podane povezave.



Slika 3.5: Arhitektura sistema

Slika 3.5 predstavlja tok podatkov med različnimi viri in nadzorno ploščo, ki podatke agregira, shranjuje in posreduje storitvi namenjeni sočasnemu oddajanju na vse povezane odjemalce preko vtičnice (*angl. socket*). Objekti, predstavljeni z modro barvo, prikazujejo elemente, ki so bili izdelani v sklopu tega diplomskega dela. Na sliki 3.5 zaradi boljše preglednosti ne prikazujem povezav znotraj posameznih modulov (primer: povezave znotraj nadzorne plošče med moduli Laravel, vrsta, Pusher SDK idr.).

3.6.1 Nadzorna plošča in viri podatkov

Na vmesnem delu oziroma delu, ki smo ga razvijali, smo pripravili takorekoč urnik opravil, ki naj se izvajajo. Določene zadeve se namreč izvajajo zgolj enkrat na dan ob določeni uri (primer: pošiljanje dnevnega poročila o uporabi sistema), medtem ko se nekatere stvari poganjajo vsako minuto delovanja (primer: pridobitev podatkov o lokacijah) do mere, ko se določen del izvede zgolj enkrat na mesec ob točno določeni uri (primer: pošiljanje mesečnega poročila o uporabi sistema). Ta napisani urnik vsako minuto poganja operacijski sistem, ki je naložen na gostujočem strežniku, in sicer govorimo o (*angl. cron*)u. To je programsko orodje, ki teče na operacijskih sistemih UNIX, namenjenih poganjanju opravil na časovni osnovi. Tako aplikacija pridobi vse podatke, povezane z glavno storitvijo. Seveda pa obstajajo še povezave na druge storitve, kot so Uptime Robot in vreme ARSO. Do vseh zunanjih storitev obravnavana rešitev dostopa preko njihovih javno dostopnih API-jev. Ob tem bi radi tudi omenili, da niso vsi podatki, ki se na strežniku obdelujejo, vezani na čas pridobivanja. Del strani je narejen tudi "na zahtevo", torej pošlje zahtevek zgolj, ko upravljalec to zahteva in želi pridobiti zelo specifičen rezultat (primer: graf, poročilo, podatke o določenem vozilu idr.).

3.6.2 Odjemalec in aplikacija

Ob vsem tem smo poskrbeli tudi za vse poglede (Odjemalec - spletna stran), do katerih dostopa upravljalec, hkrati pa ob tem dostopa do pogleda, ki

je namenjen temu, da se podatki v realnem času spreminjajo, ko pride do spremembe. Za to funkcionalnost se na začetku vzpostavi povezava na odjemalcu tudi do storitve, ki omogoča sočasno oddajanje na vse povezane odjemalce, kar posledično prinese ravno to, da ne potrebujemo nobenega posega upravljalca za osvežitev podatkov. Seveda za to potrebujemo tudi primerno prirejene komponente, ki to podpirajo, saj sama storitev ni dovolj. Tako moramo za vsako komponento, za katero želimo takšno funkcionalnost, poskrbeti zgolj za obravnavo sprejema podatkov s strani storitve, ki posreduje podatke in vedeti, kako jih je strežnik poslal, da jih nato preberemo in uporabimo. Nekatere komponente le izbrišejo stare podatke in jih nadomestijo z novimi, medtem ko nekatere podatke zgolj dodajo v svojo podatkovno strukturo.

3.6.3 Povezava storitve sočasnega oddajanja - aplikacije - odjemalca

Vsaka komponenta je bila torej sestavljena tako, da sprejme ime kanala, na katerem posluša, in ime dogodka, ki ga sprejme in navsezadnje obdela. Kanale smo razdelili na glavnega, imenovanega "dashboard" in na podkanale, ki so poleg imena "dashboard" imeli še ime regije/mesta, na katerega se je podatek navezoval. Na omenjeno storitev se aplikacija povezuje sama v trenutku, ko pride do razpršitve dogodka po celem sistemu vendar le v primeru, ko je nastavljen način sočasnega oddajanja. V primeru tega prototipa je to Pusher. Na drugi strani imamo odjemalca, ki mora najprej opraviti avtentikacijo, ki mu pove, ali ima sploh dovoljenje za poslušanje na podanem kanalu in če dobi od strežnika "zeleno luč", se nato poveže še na omenjeno storitev. Za to smo uporabili uradno knjižnico laravela, imenovano laravel-echo za odjemalca, za strežnik pa smo preko programa Composer naložili Pusher SDK. V datoteko okoljskih spremenljivk smo shranili podatke, ki se uporabijo ob vzpostavljanju povezave. To so enolični kazalnik aplikacije, ključ, skrivnost in gruča (*angl. cluster*). Vse to pridobimo ob oblikovanju aplikacije na sami storitvi. Tako se vse komponente na zaslonu ob na novo poslanem podatku s

strežnika posredujejo preko omenjene storitve na odjemalca, kjer se podatek s pomočjo Vue.js obdela in na novo upodobi pridobljene podatke na zaslonu.

Poglavje 4

Uporabljena orodja

V poglavju bomo opisali vse izbrane programe (PhpStorm, cron), storitve in ogrodja (Laravel, Vue.js), ki smo jih ob delu uporabljali.

4.1 Laravel

Odprtokodno ogrodje je zgrajeno v jeziku PHP in je namenjeno vsem, ki uživajo v dostavljanju svoje kode in produktov v tem jeziku. Lahko se pohvali s svojo dobro zasnovo, bogato in berljivo sintakso, ogromno skupnostjo, ki dnevno nadgrajuje, popravlja in za svoje delo uporablja to ogrodje. Samo ogrodje že na samem začetku v projekt prinese veliko zadev, ki se načeloma ponavljajo iz projekta v projekt in postanejo zamudne in zahtevne za vzdrževanje. To so recimo avtentikacija, avtorizacija, migracije, vrste, dogodki, pogledi, načrtovalnik izvajanja ukazov, obdelava zahtevkov in odgovorov strežnika, vmesna programska oprema (*angl. middleware*), ORM in številne druge komponente, ki jih uporablja velika večina spletnih strani oziroma aplikacij. Izmed naštetih modulov to del ne uporablja samo ORM-ja [10].

Arhitekturna zasnova ogrodja je MVC (model – view – controller), kar pomeni, da se aplikacija deli v tri logične sklope. Prvi kot **model** predstavlja obdelavo podatkov, povezavo s podatkovno bazo in vse kar je povezano s

predstavitvijo podatkov drugim ravnem aplikacije. Do njega lahko dostopa samo modul **controller**. Slednji torej skrbi za sprejem zahtevka in na osnovi zahtevka kliče zadolženi predhodni opisani modul s podanimi parametri, ki jih nato posreduje naprej zadnjemu modulu **view**, ki predstavlja to, kar uporabnik vidi. Vse skupaj si lahko predstavljamo na način, da uporabnik uporablja controller, ki poskrbi, da model pripravi pravilne podatke in jih posreduje naprej na view za prikaz le-teh [13].

Številni moduli so narejeni tako, da lahko v datoteki okoljskih spremenljivk zamenjamo gonilnik in sistem bo uporabljal popolnoma drugačno storitev, ne da bi spremenil samo aplikacijsko kodo. Vse skupaj je narejeno tako, da se uporablja načelo vmesnikov iz objektnega programiranja in tako vsi gonilniki posedujejo enake funkcije, ki imajo sebi prilagojeno kodo za pravilno delovanje. Sam programer nato uporablja razred, izpeljan iz tega vmesnika, ki ima enaka imena funkcij, kot jih imajo gonilniki in zaradi tega vse skupaj lepo deluje. Nekaj primerov takšnih modulov so vrste, podatkovna baza in predpomnjenje [9, 8, 6].

Ogrodje uporablja tudi dva standarda, in sicer PSR-2 in PSR-4. Prvi se uporablja kot standard za pisanje kode, zadnji pa nakazuje pravila samodejnega nalaganja datotek za izvajanje programa [7].

Sam ekosistem Laravela odlično podpira tudi glavni razvijalec ogrodja, in sicer v času pisanja za Laravel obstaja šest paketov in pet samostojnih aplikacij, ki še dodatno izboljšajo in pohitrijo sam razvoj aplikacij. Le-te vključujejo aplikacijo za samodejno zvezno integracijo, nadzorno ploščo za prioritetno vrsto, mikro-ogrodje kot okrnjeno in hitrejšo različico Laravela, namenjeno aplikacijam za izdelavo vmesnikov API, administracijsko ploščo za urejanje vsebine spletne strani in aplikacijo, ki služi za hitro vzpostavitev plačilnega sistema in nadzora uporabnikov [5].

4.2 PhpStorm

Program PhpStorm je eden izmed programov iz "družine" IntelliJ češkega podjetja JetBrains, z vsemi mogočimi dodatki, ki razvijalcu omogočajo hitrejši in lažji razvoj spletnih aplikacij. To je nekako nadgradnja programa WebStorm, ki enako podpira razvoj spletnih aplikacij, le da ta k naboru jezikov, kot sta HTML in JavaScript, dodaja še skriptni jezik PHP in jezik za urejanje podatkovnih baz SQL. Za uporabo potrebujemo licenco, ki obstajajo v več primerih. Prvi je seveda, da jo kupimo, obstajata pa še dva načina, kako jo dobiti brezplačno. Kot študentu, učitelju ali raziskovalcu nam pripada licenca, veljavna 365 dni. Za pridobitev le-te potrebujemo univerzitetni e-naslov ali e-naslov s končnico ".edu". Tretja možnost je sodelovanje v njihovem tako imenovanem programu "EAP", kjer dobimo popolno različico programa, vendar obstaja možnost programskih napak in občasnih sesutij programa, s tem pa pomagamo pri odkritju le-teh in izboljšanju končne različice programa [15].

4.3 Cron

Cron je program oziroma orodje, namenjeno izvajanju vseh časovno povezani opravil. Najdemo ga na operacijskih sistemih UNIX. Najbolj primeren je za izvajanje ponovljivih nalog, ki so tipično povezane z administracijo in vzdrževanjem sistema. Za svoje delovanje uporablja datoteko, imenovano *crontab*. V slednji je vsak ukaz predstavljen v svoji vrsti in sestavljen iz šestih delov. S prvimi petimi opišemo časovno odvisnost, kdaj naj se opravilo izvaja, z zadnjim pa podamo ukaz ali zaporedje ukazov, ki naj se izvedejo [11].

4.4 Vue.js

Vue.js je ogrodje JavaScript, čigar največja lastnost po našem mnenju je progresivnost oziroma postopnost. Omogoča namreč, da ogrodje vključimo

v zgolj majhen delček strani, kot je recimo ena komponenta, pa naj bo to gumb, tabela ali celo galerija in pripelje celo do tega, da lahko samo z uporabo ogrodja izdelamo celotno spletno stran. Za slednji podvig imamo na voljo že pakete, ki jih podpirajo izdelovalci in omogočajo usmerjanje, nadzor stanja in podatkov ter celo upodabljanje (izris) strani na strežniku, kljub temu da je to ogrodje JavaScript. Vue.js utilizira virtualni DOM za svoje delovanje in spreminjanje vsebine. Kot ogrodje je popolnoma reaktivno in brez stanja. Fokus razvoja je na jedru ogrodja, ki je, kolikor se da preprosto, za kompleksnejše zadeve pa nato poskrbijo dodatki, kot jih omenjamo zgoraj (primer: usmerjanje) [16].

4.5 Pusher

Pusher je storitev, ki omogoča izmenjavo podatkov v stvarnem času med vsemi odjemalci, naročenimi na določen kanal. V našem primeru strežnik ob oddajanju dogodka pošlje vse podatke dogodka na Pusher, ki vse te podatke v obliki JSON posreduje vsem odjemalcem, ki poslušajo na kanalu, kjer je strežnik oddal dogodek. To pa ni edina storitev, ki jo Pusher ponuja. Med drugim ponujajo tudi pošiljanje potisnih obvestil na mobilne naprave in v času pisanja imajo v testiranju tudi svoje ogrodje za izdelavo klepetalnic [12].

Poglavje 5

Zaslonske maske in prikaz delovanja

Skozi celotno poglavje smo uporabljali uporabnika iz skupine "Vodstvo", tako da bo viden celoten obseg spletne aplikacije.

5.1 Prvi del rešitve

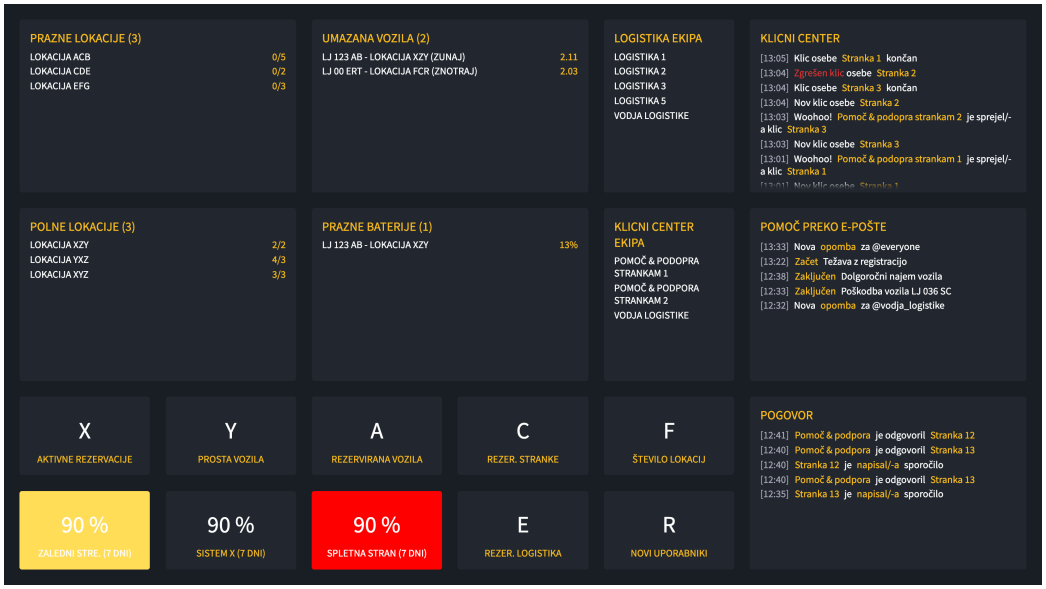
5.1.1 Vstopna stran

Vstopna stran je namenjena pregledu na napravah, ki nam omogočajo vnos podatkov. Za prikaz na televiziji bi uporabljali Raspberry PI, ki že ima prednastavljeno prvo stran. Na sliki 5.1 upravljalec, ki se prijavi, vidi vse module, do katerih ima dodeljen dostop.

Slika 5.1: Izgled vstopne strani aplikacije

5.1.2 Osnove logistike in pomoči strankam

To je bila stran (slika 5.2), ki je bila narejena prva, in sicer kot nekakšen dokaz koncepta. Na njej so vse osnovne stvari o trenutnem delovanju, ki so lahko v pomoč vsem zaposlenim v logističnem delu ekipe oziroma na terenu in osebam, zaposlenim v oddelku za pomoč strankam, kjer jim odgovarjajo na vsa vprašanja preko telefona, e-pošte in klepetalnika.



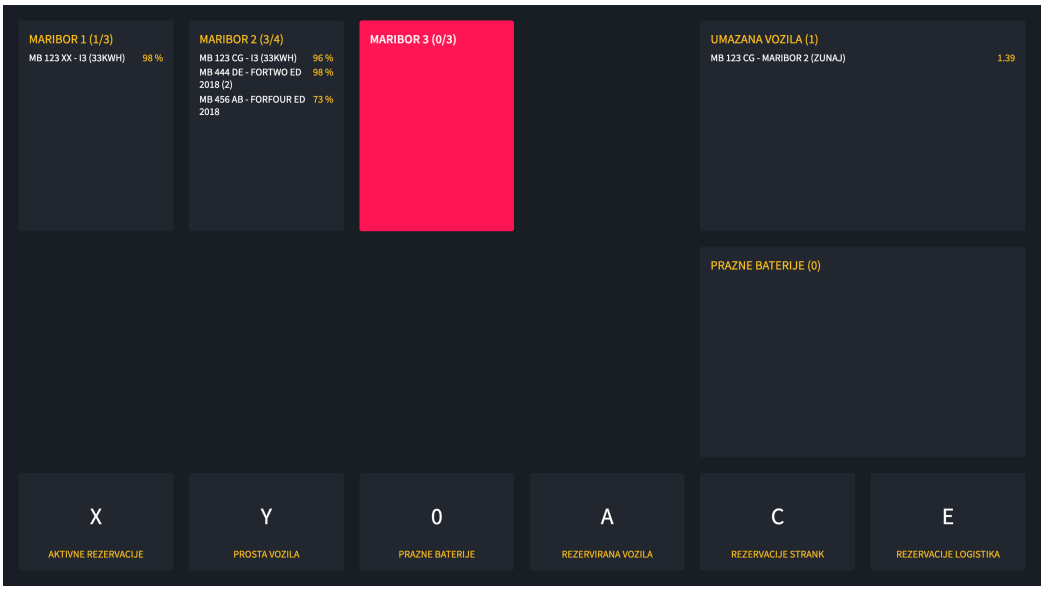
Slika 5.2: Izgled osnovne strani za logistiko in pomoč uporabnikom

5.1.3 Primerjava večje in manjše regije v sistemu souporabe vozil

Predstavljeni sta dve regiji, ena z veliko lokacijami, ki predstavlja večjo regijo in druga kot malce manjša regija v okviru samega sistema.

Slika 5.3: Izgled večje regije

Zaradi same številčnosti lokacij v večji regiji (slika 5.3) je na strani uporabljen kompakten videz lokacij, ki nakazuje samo stanje lokacije (število vozil na lokaciji/število parkirnih mest na lokaciji).

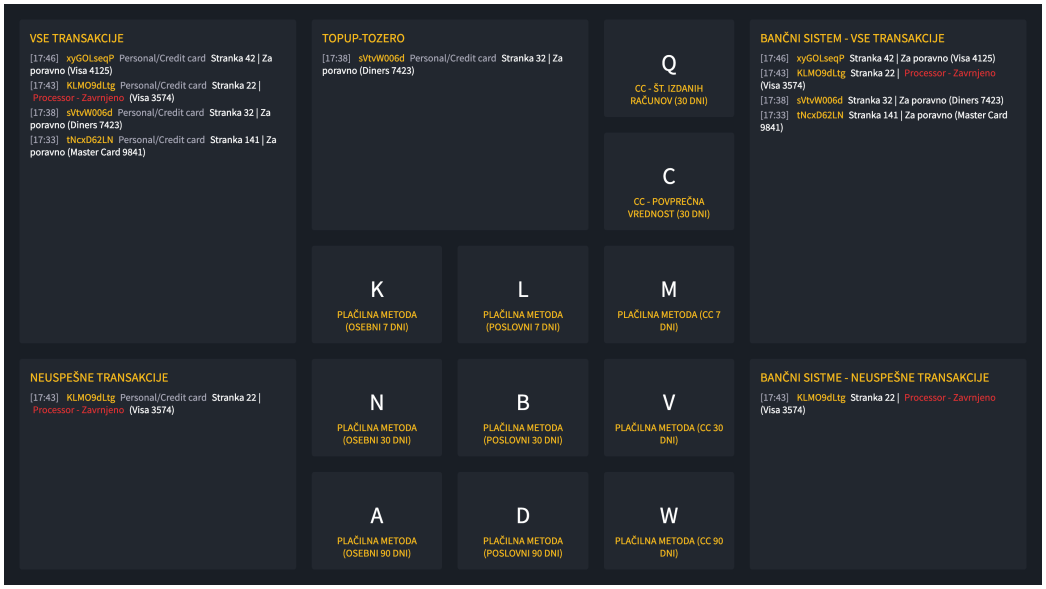


Slika 5.4: Izgled manjše regije

Medtem je pri manjši regiji (slika 5.4) pogled lokacije veliko bolj bogat in prikazuje še podrobnosti vozila, kot so model, registrska številka in raven električne energije.

5.1.4 Opravljene transakcije

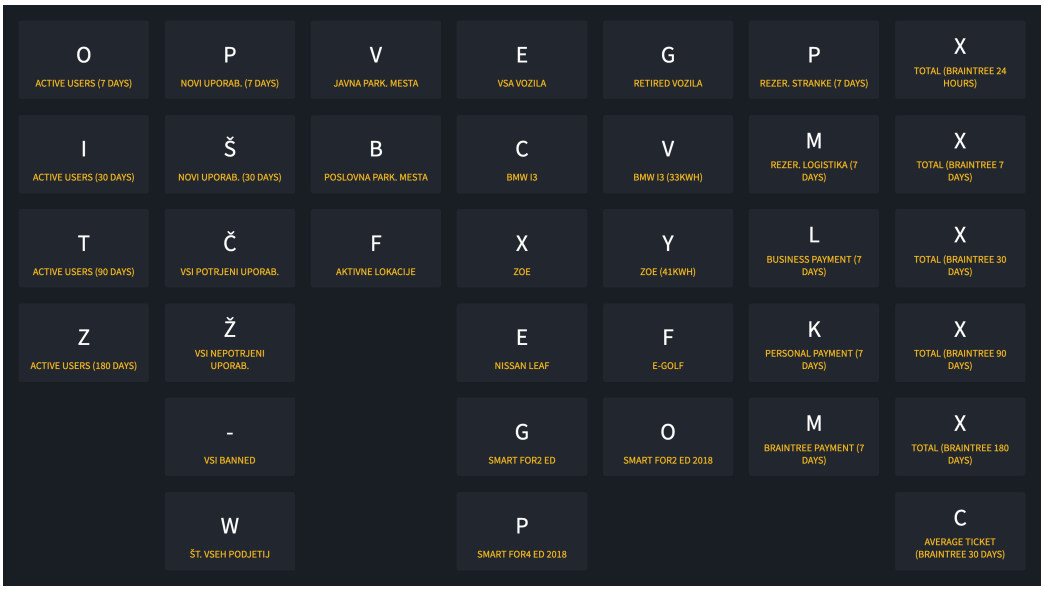
Živi tok podatkov predstavlja vse zaključene transakcije v sistemu in omogoča računovodstvu hitro ukrepanje ob neuspešnih transakcijah. Ob tem je podanih še nekaj številk o razlikah v tipih transakcij, kot je vidno na sliki 5.5.



Slika 5.5: Izgled strani za spremljanje transakcij

5.1.5 Ključni kazalniki uspešnosti

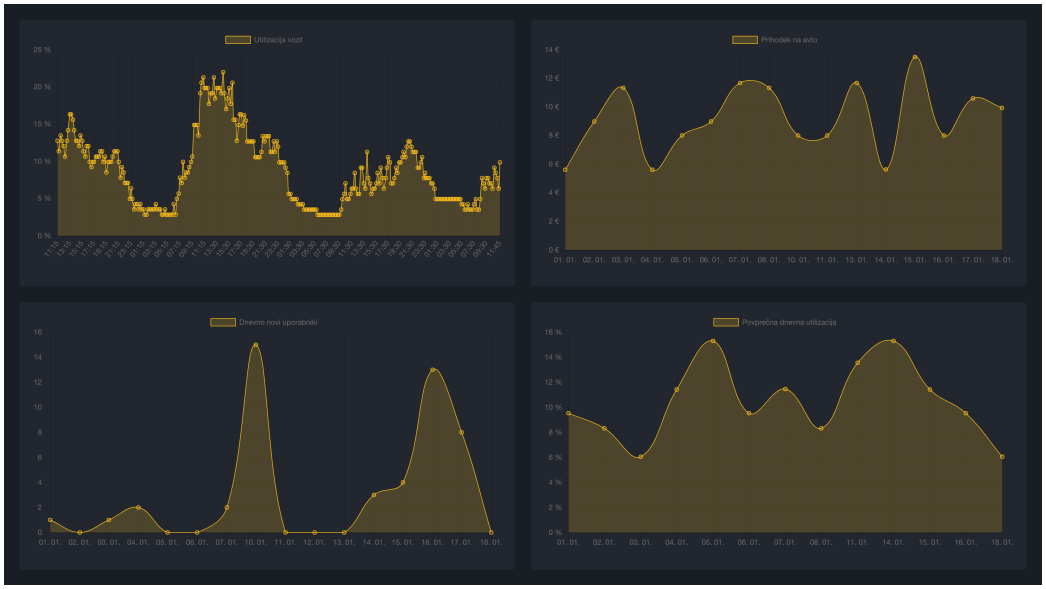
Ključni kazalniki uspeha lahko direktorju oziroma vodstvu podjetja hitro prikažejo trenutno sliko sistema kot dejstva, predstavljena s številkami, vidna na sliki 5.6.



Slika 5.6: Izgled strani s podanimi kazalniki uspešnosti

5.1.6 Grafi

Slika 5.7 prikazuje primer grafov za vodstvo. Graf utilizacije se osveži vsakih 15 minut, medtem ko se ostali osvežijo konec dneva.

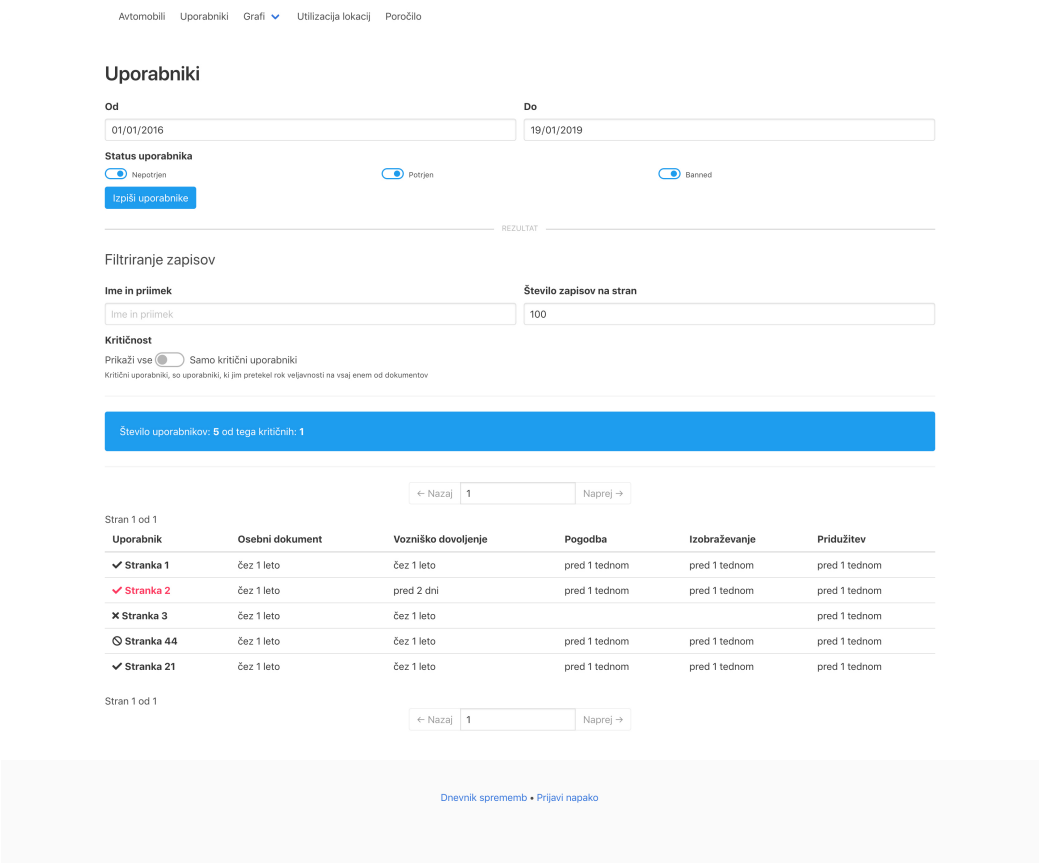


Slika 5.7: Izgled strani z grafi kot kazalniki uspešnosti in trenda

5.2 Drugi del rešitve

5.2.1 Uporabniki in vozila

Za enostaven pregled registriranih uporabnikov in aktivnih vozil v sami floti sistema smo pripravili grafični vmesnik, ki olajša iskanje in nadzor nad podanimi viri.



Slika 5.8: Izgled strani za pregled uporabnikov sistema

Slika 5.8 prikazuje izpisane uporabnike sistema, o katerih vidimo zgolj najbolj osnovne podatke (veljavnost vozniškega dovoljenja in osebnega dokumenta, registracija v sistem idr.). Enak pogled je narejen za vozila, kjer so prikazani zgolj drugi atributi in filtri po podanih atributih.

5.2.2 Grafi

Pri izpisu vsakega grafa imamo na razpolago izbire časovnega območja, za katerega želimo pregledati agregirane podatke. Na tem sklopu podatkov nato vključimo še dodatne filtre, ki lahko vključijo ali izključijo določen tip opravljene souporabe vozila. Sledi še časovni interval, preko katerega lahko določimo, kako natančen graf želimo izpisati. Izbiramo lahko med možnostmi,

kot so dan, ura, mesec in leto. Glede na izbrano možnost sistem nato naredi intervale, v katere porazdeli opravljene najeme. Grafi imajo prav tako še možnost izbire, ali želimo kot časovni parameter uporabiti začetek ali konec najema. Kot zadnjo možnost lahko še izberemo, ali želimo vse najeme obravnavati enako ali pa za vsak tip najema želimo svojo krivuljo/stolpec.

Poglavje 6

Možnosti nadaljnjega razvoja

V tem poglavju smo opisali, na kakšen način se lahko aplikacija še razširi v smislu samega tehnološkega razvoja in poslovne domene projekta.

6.1 Tehnični razvoj

Kar se tiče samega tehničnega razvoja, bi lahko aplikacijo izboljšali tako, da bi bila primerna tudi za mobilne naprave oziroma manjše zaslone. Trenutno se uporablja precej nov način pozicioniranja v samem CSS-ju z uporabo funkcije grid, ki pa še ni tako dobro podprta, prav tako pa se pojavi težava tudi s tem, da se vsebina za majhne zaslone ne spremeni dovolj, ob podani tehnični zahtevi, da je vedno vidna vsa vsebina na velikem zaslonu brez posega upravljalca Ta del se tiče zgolj same nadzorne plošče, ne pa tudi drugega dela, namenjenega nadzoru in generiranju poročil.

Sama vrsta je implementirana s pomočjo podatkovne baze SQLite, kamor se v eno tabelo zapisujejo vsa čakajoča opravila, ki jih nato sistem izvaja zaporedoma, zato bi se na tem mestu lahko uporabila bolj namenska tehnologija, recimo strežnik redis ali pa kakšen drug MQ (RabbitMQ, IronMQ). V primeru uporabe strežnika redis bi lahko nanj premaknili tudi sistem za razpršeno oddajanje, ker trenutno uporabljamo Pusher.

Uporabniki so implementirani kot skupine, torej nima vsak upravljalec

svojega uporabniškega imena in gesla. Kot nadgradnjo bi lahko v primeru, če podjetje uporablja Active Directory ali drugo podobno storitev, le-to uporabili kot avtentikacijsko in avtorizacijsko plast.

Samo aplikacijo bi lahko skalirali vertikalno in imeli namenske strežnike za izvajanje določenih opravil v primeru visokega prometa oziroma opravljanja kakšnih preformančno zahtevnejših opravil.

6.2 Razvoj v sklopu poslovne domene

Sam sistem souporabe vozil je v zgodnji fazi razvoja zato se veliko stvari še razvija in postopoma dodaja v uporabo, s čimer se razvija tudi naša nadzorna plošča. Z novimi potrebami in lastnostmi sistema se razvija tudi sama aplikacija.

Povratna informacija uporabnikov na različnih ravneh bo predstavljala osnovni vnos za naslednje korake na področju sprememb v okviru poslovne domene. V primeru skalabilnosti celotne storitve oziroma omogočanja licenčnega dajanja sistema v uporabo, je nadzorna plošča lahko dodatni vidik monetizacije storitve.

Poglavje 7

Zaključek

Namen diplomskega dela je bil izdelati orodje, ki omogoča enostavno in učinkovito spremljanje poslovnih procesov projekta souporabe vozil. Tega smo se lotili tako, da smo najprej naredili načrt zasnove in uporabe aplikacije, opisali tehnologije, ki so pomagale pri implementaciji in premagovanju tehničnih zahtev. Namen dela je uresničen, saj nam končni proizvod omogoča enostavno in učinkovito spremljanje poslovnih procesov na vseh ravneh z upoštevanimi tehničnimi omejitvami. Za konec smo pregledali in opredelili možnosti nadaljnjega razvoja s stališča tehničnega razvoja in poslovne domene.

Literatura

[1] Avant car d.o.o. Avant2go. Dosegljivo: https://avant2go.com/. [Dostopano: 14. 10. 2019].

[2] Avant car d.o.o. Predstavitev projekta avant2go. Interni dokument, 2016. [Dostopano 08. 01. 2019].

[3] Monitor Deloitte. Car sharing in europe — business models, national variations and upcoming disruptions. Dosegljivo: https://www2.deloitte.com/content/dam/Deloitte/de/Documents/consumer-industrial-products/CIP-Automotive-Car-Sharing-in-Europe.pdf, 2017. [Dostopano: 09. 01. 2019].

[4] Tomaž Kogoj. Povezava transformacijskega vodenja z inovativnostjo: primer projekta souporabe električnih vozil avant2go. Magistrsko delo, Ekonomska fakulteta, Univerza v Ljubljani, 2018.

[5] Laravel. Laravel. Dosegljivo: https://laravel.com/. [Dostopano: 09. 01. 2019].

[6] Laravel. Laravel - cache. Dosegljivo: https://laravel.com/docs/5.7/cache#configuration. [Dostopano: 20. 01. 2019].

[7] Laravel. Laravel - contribution guide. Dosegljivo: https://laravel.com/docs/5.7/contributions. [Dostopano: 20. 01. 2019].

[8] Laravel. Laravel - database. Dosegljivo: https://laravel.com/docs/5.7/database#introduction. [Dostopano: 20. 01. 2019].

[9] Laravel. Laravel - queues. Dosegljivo: https://laravel.com/docs/5.7/queues#driver-prerequisites. [Dostopano: 20. 01. 2019].

[10] Laravel. Laravel dokumentacija. Dosegljivo: https://laravel.com/docs/5.7. [Dostopano: 09. 01. 2019].

[11] SK OSTechNix. A beginners guide to cron jobs. Dosegljivo: https://www.ostechnix.com/a-beginners-guide-to-cron-jobs/. [Dostopano: 20. 01. 2019].

[12] Pusher. Pusher. Dosegljivo: https://pusher.com/. [Dostopano: 14. 10. 2018].

[13] Karen Scarfone, Brittany Storoz, Chris Mills, Markg, Geoffroy Bailly, misaal9, Rayann Tedds, John Whitlock, and Leon Earl. The theory behind model view controller. Dosegljivo: https://developer.mozilla.org/en-US/docs/Web/Apps/Fundamentals/Modern_web_app_architecture/MVC_architecture. [Dostopano: 18. 10. 2019].

[14] Spatie. dashboard.spatie.be. Dosegljivo: https://github.com/spatie/dashboard.spatie.be. [Dostopano: 23. 01. 2019].

[15] JetBrains s.r.o. Phpstorm. Dosegljivo: https://www.jetbrains.com/phpstorm/. [Dostopano: 14. 10. 2018].

[16] Vuejs.org. Dokumentacija vue.js. Dosegljivo: https://vuejs.org. [Dostopano: 14. 10. 2018].